OMSI 2000 FORTRAN

September 15, 1973 Manual

(Pre-Publication Proof)

AUTHORS:

DAN CALEF
DEAN FASSNIDGE
MARK HORTON
DAVE MENICOSY

OMSI 2000 FORTRAN IS AN INTERPRETIVE FORTRAN LANGUAGE SIMULATOR DESIGNED TO RUN ON A HEWLETT-PACKARD 2000B, 2000C, 2000E, OR 2000F TIME-SHARED BASIC COMPUTER.

COPYRGHT 1973 BY OMSI
THE OREGON MUSEUM OF SCIENCE AND INDUSTRY
4015 SW CANYON ROAD
PORTLAND, OREGON 97221

ALL RIGHTS RESERVED PRINTED IN USA

OMSI 2000 FORTRAN

ACNOWLEDGEMENTS

THE AUTHORS OF OMSI 2000 FORTRAN WOULD LIKE TO EXTEND THEIR THANKS TO MR. CARLTON BRYSON AND MR. HARLAN HENRY, MATHEMATICS TEACHERS AT BENSON HIGH SCHOOL FOR THEIR TIRELESS ASSISTANCE AND ENCOURAGEMENT. OMSI 2000 FORTRAN WAS DEVELOPED WITH THE COOPERATION OF AND THE ENCOURAGEMENT OF THE MULTNOMAH COUNTY INTERMEDIATE EDUCATIONAL DISTRICT.

REPRODUCING THE OMSI 2000 FORTRAN MANUAL

THIS MANUAL MAY BE REPRODUCED IN ITS ENTIRETY BY REGISTERED SUBSCRIBERS TO OMSI 2000 FORTRAN SOFTWARE WITHOUT SPECIFIC AUTHORIZATION FROM OMSI IF IT BEARS A STATEMENT IN PLACE OF THIS PARAGRAPH TO THE EFFECT THAT, "THIS MANUAL IS REPRODUCED IN ITS ENTIRETY BY [REGISTERED USER, TITLE] OF [ORGANIZATION, CITY, STATE] ON [MONTH, DAY, YEAR] BY AGREEMENT WITH THE OREGON MUSEUM OF SCIENCE AND INDUSTRY, PORTLAND, OREGON.

REPRODUCTION OR REDISTRIBUTION OF THIS MANUAL OR OMSI 2000 FORTRAN SOFTWARE IN PART OR IN WHOLE WITHOUT THE PRIOR WRITTEN PERMISSION OF OMSI IS FORBIDDEN.

HEWLETT-PACKARD AND HP ARE TRADEMARKS OF HEWLETT-PACKARD, CUPERTINO, CALIFORNIA.

BASIC IS A COMPUTER LANGUAGE CREATED AT DARTMOUTH COLLEGE.

PREFACE

OMSI 2000 FORTRAN WAS DESIGNED WITH THE THOUGHT THAT STUDENTS HAVING ACCESS TO ONLY A HEWLETT-PACKARD 2000B, 2000C, 2000E, OR 2000F TIME-SHARED BASIC COMPUTER SHOULD HAVE FACILITIES TO LEARN OTHER COMPUTER LANGUAGES. USING THE FORTRAN PROGRAM PACKAGE, THE STUDENT CAN, INDIVIDUALLY OR IN A GROUP, LEARN A SECOND COMPUTER LANGUAGE WITHOUT EXPENSIVE HOOK-UPS.

OMSI 2000 FORTRAN IS AN INTERPRETIVE FORTRAN LANGUAGE SIMULATOR WRITTEN IN HP 2000 BASIC. IN WRITING THIS PROGRAM, WE HAVE PLACED MORE STRESS ON THE ACCURACY OF THE SIMULATION RATHER THAN THE OPERATIONAL SPEED. IF IT SEEMS TO TAKE A LONG TIME TO EXECUTE A PROGRAM, PLEASE KEEP IN MIND THE METHOD AND COMPLEXITY OF THE TASK.

THE FORTRAN MANUAL IS DESIGNED TO PROVIDE A DESCRIPTION OF THE FEATURES AVAILABLE IN THE FORTRAN PACKAGE, AND IS NOT INTENDED AS AN INSTRUCTIONAL MANUAL. A STUDENT GUIDE AND INSTRUCTORS HANDBOOK ARE AVAILABLE SEPARATELY. THE FIRST SECTION OF THE MANUAL DESCRIBES THE FEATURES AVAILABLE IN THE 2000B/E VERSION OF FORTRAN; ADDITIONAL FEATURES AVAILABLE IN THE 2000C/F VERSION OF FORTRAN ARE DESCRIBED IN THE SECOND SECTION.

—THE AUTHORS

TO REPORT BUGS:

THE AUTHORS WISH TO RECIEVE REPORTS OF BUGS IN OMSI 2000 FORTRAN. THE AUTHORS ARE PARTICULARLY EAGER TO HEAR FROM USERS ABOUT BUGS WHICH DEMONSTRATE OMSI 2000 FORTRAN'S DEPARTURE FROM STANDARD FORTRAN CONVENTION OR WHICH CAUSE UNEXPECTED EXITS FROM FORTRAN BACK TO BASIC.

PLEASE CREATE A "SIMPLEST CASE" ILLUSTRATION OF THE BUG, THEN CREATE A TELETYPE RECORD OF THE PROBLEM STARTING WITH 'GET-FORTRN'. WRITE ON THE TELETYPE OUTPUT IF EXPLANATORY REMARKS ARE NEEDED. WHEN A LENGTHY PROGRAM MUST BE USED, ATTEMPT TO INCLUDE A PAPER TAPE OF IT.

BE SURE TO INCLUDE YOUR NAME, ADDRESS AND PHONE NUMBER, AND SEND BUG REPORTS TO:

OMSI 2000 FORTRAN
SOFTWARE DEVELOPMENT GROUP
OREGON MUSEUM OF SCIENCE AND INDUSTRY
4015 SW CANYON ROAD
PORTLAND, OREGON 97221

FULLY DOCUMENTED BUG REPORTS WILL BE ACKNOWLEDGED.

TABLE OF CONTENTS

SECTION 1 B/E FORTRAN

SYSTEM OPERATION 1-1
PROGRAM CONSTRUCTION 1-2
PROGRAM EXECUTION 1-3
VARIABLES 1-4

COMMANDS

EXECUTE, TAPE, LOAD, CONTROL Y 1-4
LIST, TPUNCH, UJOB, KJOB, HELP 1-5
STORE, RETRIEVE, ERASE, CATALOG 1-6

STATEMENTS

ASSIGNMENT STATEMENT, READ 1-7
WRITE, GO TO, DO, CONTINUE 1-8
FORMAT 1-9
STOP, IF, END, COMMENT 1-10
FUNCTIONS 1-11
EDITING 1-12

SECTION 2 C/F EXTENSIONS

COMMANDS

LN-M,N, OPEN, REASSIGN, PUNCH 2-1

STATEMENTS

REAL AND INTEGER, LOGICAL IF 2-2
ADDITIONAL FUNCTIONS, PAUSE 2-3
DATA, FORMAT 2-4
COMPUTED GO TO,
SUBSCRIPTED VARIABLES 2-5
DATA FILES 2-6

SECTION 3 APPENDIX

FORTRAN FILE STRUCTURE 3-1
SPECIAL CHARACTERS 3-2
2000B/E TAPE LOADING INSTRUCTIONS 3-3
2000B/E SYSTEM LIBRARY USE 3-3
2000C/F TAPE LOADIING INSTRUCTIONS 3-4
2000C/F SYSTEM/GROUP LIBRARY USE 3-4

ERROR MESSAGES

SYNTAX AND PROGRAM CONSTRUCTION 3-5
EXECUTION 3-7

SECTION 4 DEMONSTRATION PROGRAMS

TABLE OF CONTENTS

SECTION 1 B/E FORTRAN

SYSTEM OPERATION 1-1
PROGRAM CONSTRUCTION 1-2
PROGRAM EXECUTION 1-3
VARIABLES 1-4

COMMANDS

EXECUTE, TAPE, LOAD, CONTROL Y 1-4
LIST, TPUNCH, UJOB, KJOB, HELP 1-5
STORE, RETRIEVE, ERASE, CATALOG 1-6

STATEMENTS

ASSIGNMENT STATEMENT, READ 1-7
WRITE, GO TO, DO, CONTINUE 1-8
FORMAT 1-9
STOP, IF, END, COMMENT 1-10
FUNCTIONS 1-11
EDITING 1-12

SECTION 2 C/F EXTENSIONS

COMMANDS

LN-M,N, OPEN, REASSIGN, PUNCH 2-1

STATEMENTS

REAL AND INTEGER, LOGICAL IF 2-2
ADDITIONAL FUNCTIONS, PAUSE 2-3
DATA, FORMAT 2-4
COMPUTED GO TO,
SUBSCRIPTED VARIABLES 2-5
DATA FILES 2-6

SECTION 3 APPENDIX

FORTRAN FILE STRUCTURE 3-1
SPECIAL CHARACTERS 3-2
2000B/E TAPE LOADING INSTRUCTIONS 3-3
2000B/E SYSTEM LIBRARY USE 3-3
2000C/F TAPE LOADIING INSTRUCTIONS 3-4
2000C/F SYSTEM/GROUP LIBRARY USE 3-4

ERROR MESSAGES

SYNTAX AND PROGRAM CONSTRUCTION 3-5
EXECUTION 3-7

SECTION 4 DEMONSTRATION PROGRAMS

SYSTEM OPERATION

TO OPERATE OMSI 2000 FORTRAN, THE USER MUST OPEN A SET OF FILES. THESE FILES MUST BE AVAILABLE ON A READ/WRITE BASIS. THE FILE NAMES AND SIZES ARE AS FOLLOWS:

NAME	
FTNFLS	10 RECORDS
FTNULT	8 RECORDS
FTNCOP	48 RECORDS

THESE FILE SIZES ARE FOR A SYSTEM WHICH CAN RUN 48 STEP PROGRAMS. IF THE SYSTEM IS ONLY GOING TO RUN SHORT PROGRAMS, THE FILES FTNFLS AND FTNCOP CAN BE DECREASED IN SIZE ACCORDINGLY. FTNCOP REQUIRES APPROXIMATELY ONE RECORD PER STATEMENT. FOR MORE INFORMATION ON FILE SIZES, REFER TO PAGE 3-1.

AFTER THE FILES HAVE BEEN OPENED, THE USER MUST RETRIEVE FORTRAN. THIS IS ACCOMPLISHED BY TYPING 'GET-FORTRN' (OR 'GET-$FORTRN' OR 'GET-*FORTRN', DEPENDING UPON THE LIBRARY IN WHICH IT RESIDES). THE USER THEN RUNS FORTRN, AND THE SYSTEM RESPONDS WITH A HEADING (OMSI 2000 FORTRAN) OR WITH A FILE STATUS ANNOUNCEMENT. MISSING OR PROTECTED FILE MEANS THAT THE FILES HAVE NOT BEEN OPENED PROPERLY. CHECK FOR MISSPELLED NAMES. READ ONLY FILES: MEANS THAT THE FILES ARE BUSY, AND UNLESS THE USER IS IN AN AXXX ACCOUNT, HE WILL NOT BE ABLE TO RUN THE PACKAGE. IF THE SYSTEM FILES ARE AVAILABLE, THE SYSTEM PRINTS A PERIOD IN THE LEFT-MOST COLUMN, AND IS READY TO ACCEPT A COMMAND.

THE FORTRAN SIMULATOR IS UNABLE TO TAKE ACTION ON ANY INPUT UNTIL THE USER TYPES A CARRIAGE RETURN. THE 'BREAK' KEY MAY BE USED ONLY WHILE OMSI 2000 FORTRAN IS EXECUTING A FORTRAN PROGRAM, AS IF IT IS PRESSED AT ANY OTHER TIME, CERTAIN FILE DATA MAY BE INADVERTENTLY DESTROYED.

BEFORE THE B/E VERSION OF OMSI 2000 FORTRAN CAN RUN ON A 2000B, THE USER MUST REMOVE ALL QUOTATION MARKS FROM THE CHAIN STATEMENTS. TO DO SO, REFER TO THE TABLE OF CHAIN STATEMENTS ON PAGE 3-3. GET THE PROGRAM, TYPE THE INDICATED LINES AS THEY ARE LISTED WITHOUT TYPING THE QUOTATION MARKS, AND THEN KILL AND CSAVE THE PROGRAM. IF FORTRAN IS NOT
SYSTEM LIBRARY, THE DOLLAR SIGNS ($) MUST BE OMITTED. FOR EXAMPLE:

```
USE IN THE SYSTEM LIBRARY      USE IN ANY OTHER ACCOUNT
GET-FORTRN                     GET-FORTRN
320  CHAIN $FTNLIB             320  CHAIN FTNLIB
1000  CHAIN $FTNINS            1000  CHAIN FTNINS
1130  CHAIN $FTNCPL            1130  CHAIN FTNCPL
KILL-FORTRN                    KILL-FORTRN
SAVE                           SAVE
AND SO ON                      AND SO ON
```

IF THE SPACE IS AVAILABLE, IT IS RECCOMENDED THAT THE OMSI 2000 FORTRAN PROGRAMS BE CSAVED, AS THEY CHAIN FREQUENLY.

PROGRAM CONSTRUCTON

IN FORTRAN, LINE NUMBERS ARE NOT REQUIRED AND ARE USED FOR INTERNAL PROGRAM REFERENCE ONLY. THEY ARE NOT USED FOR PROGRAM SEQUENCING. THE FIRST FIVE COLUMNS OF EACH STATEMENT ARE RESERVED FOR LINE NUMBERS, AND THE STATEMENT MUST BEGIN IN COLUMN SEVEN. TYPING A CONTROL I (^I) WILL CAUSE THE SYSTEM TO MOVE THE PORTION OF THE STATEMENT AFTER THE CONTROL I TO THE SEVENTH COLUMN.

```
EXAMPLE:        12345678  (COLUMN NUMBER)

                 CONTROL I
THE USER TYPES: READ(1,100)ALPHA,BETA
WHICH LISTS AS:       READ(1,100)ALPHA,BETA

                    CONTROL I
THE USER TYPES: 100FORMAT(X,2F10.5)
WHICH LISTS AS: 100   FORMAT(X,2F10.5)
```

OMSI 2000 FORTRAN REQUIRES THAT THE PROGRAM BE STRUCTURED IN A CERTAIN MANNER SIMILAR TO THAT USED BY A LARGE FORTRAN COMPILER. SPECIFICALLY, ALL NON-EXECUTABLE STATEMENTS (FORMAT, AND IF AVAILABLE, INTEGER, REAL, DATA, DIMENSION, AND DEFINE FILE) MUST BE PLACED BEFORE ANY EXECUTABLE STATEMENT. IF A DIMENSION STATEMENT IS USED, IT MUST PRECEED ANY OTHER NON-EXECUTABLE STATEMENT. APPEARANCE OF A NON-EXECUTABLE STATEMENT AFTER AN EXECUTABLE STATEMENT WILL CAUSE A FATAL ERROR. NON-EXECUTABLE STATEMENTS CANNOT BE REFERENCED BY DO, GOTO, OR IF STATEMENTS. THE FINAL STATEMENT IN THE PROGRAM MUST BE AN END STATEMENT, AND ANYTHING APPEARING AFTER IT IS ASSUMED TO BE DATA. HENCE, ONLY ONE END IS ALLOWED, AND ALL DATA MUST FOLLOW DIRECTLY AFTER IT.

PROGRAM EXECUTION

FOR THE BEGINING USER, THE FOLLOWING SECTION DESCRIBES HOW TO EXECUTE A SIMPLE FORTRAN PROGRAM USING THE SIMULATOR. THE FULL SET OF COMMANDS IS EXPLAINED FURTHER IN THE MANUAL.

AFTER THE USER HAS RETRIEVED THE SYSTEM AND IS EXECUTING IT, THE SYSTEM RESPONDS WITH A PERIOD, LIKE THIS:

```
.                             THE USER WILL PROBABLY WANT TO
                              LOAD A FORTRAN PROGRAM.  TO DO SO, HE
                              TYPES 'LOAD'.  THE SYSTEM RESPONDS:
.LOAD
PROGRAM NAME--X               THE USER TYPES THE NAME OF HIS PROGRAM, "X",
                              WHICH IS LIMITED TO TEN PRINTING CHARACTERS,
                              AND THEN HIS PROGRAM, USING A CONTROL I (^I)
                              TO TAB TO COLUMN SEVEN.  WHEN THE USER WISHES
^Y                            TO RETURN TO THE MONITOR, HE TYPES A CONTROL
                              Y (^Y), FOLLOWED BY A RETURN.  THE SYSTEM
EXIT                          WILL PRINT "EXIT", INDICATING A RETURN TO THE
                              MONITOR.  AT THIS POINT, THERE WILL USUALLY
                              BE ERRORS THE USER WISHES TO CORRECT.  HE
.                             DOES SO BY TYPING THE APPROPRIATE EDITING
                              COMMANDS (DESCRIBED FURTHER ON IN THE MANUAL),
                              OR EXECUTES HIS PROGRAM BY TYPING 'EXECUTE'.
.EXECUTE                      THE SYSTEM THEN COMPILES THE PROGRAM, AND
                              NOTIFIES THE USER BY TYPING:
LOADING    FORTRAN 'X'        WHERE X IS THE PROGRAM NAME.  AT THE END OF
                              COMPILATION, THE SYSTEM TYPES:
EXECUTION  FORTRAN 'X'        AND THEN EXECUTES THE PROGRAM.  IF THE PROGRAM
                              RUNS SUCCESSFULLY TO COMPLETION, THE
EXECUTION COMPLETED           MESSAGE IS PRINTED OUT.  THE USER IS THEN
.                             RETURNED TO THE MONITOR.
```

VARIABLES

IN FORTRAN, VARIABLE NAMES CAN BE FROM ONE TO SIX CHARACTERS LONG. THE FIRST CHARACTER MUST BE A LETTER, ANY OTHER CHARACTERS MAY BE EITHER LETTERS OR DIGITS. THE USER SHOULD AVOID VARIABLE NAMES THAT ARE THE NAMES OF FUNCTIONS OR CONTAIN STATEMENT NAMES. VARIABLES ARE OF TWO TYPES: INTEGER VARIABLES, THOSE STARTING WITH THE LETTERS I, J, K, L, M, OR N, OR REAL (FLOATING POINT) VARIABLES, WHICH START WITH ANY OTHER LETTER.

EXAMPLES:

VARIABLE NAME	VALID OR INVALID	TYPE
ALPHA	VALID	REAL
KEGG	VALID	INTEGER
INT	INVALID; 'INT' IS A FUNCTION NAME	
BAD46	VALID	REAL
MAN-45	INVALID; '-' IS A SPECIAL CHARACTER	
ZAPPA	VALID	REAL
DOG	INVALID; 'DO' IS THE NAME OF A STATEMENT	
ABCDEFG	INVALID; TOO MANY CHARACTERS.	

EXECUTE

PURPOSE: THIS COMMAND STARTS THE EXECUTION OF THE FORTRAN PROGRAM FROM THE FIRST STATEMENT TO THE LAST.

INPUT: AFTER A PROGRAM HAS BEEN LOADED, AND THE USER IS OUT OF THE LOAD MODE, A PERIOD IS PRINTED. THE USER THEN TYPES 'EXECUTE'. THE SYSTEM PRINTS "LOADING FORTRAN 'PROGRAM NAME'", AND LATER PRINTS "EXECUTION FORTRAN 'PROGRAM NAME'". AT THIS TIME, THE PROGRAM WILL BE EXECUTED, AND ANY EXECUTION ERRORS WILL BE REPORTED TO THE USER, FOLLOWED BY THE "EXECUTION ABORTED" ERROR MESSAGE. IF THE PROGRAM RUNS TO COMPLETION, THE SYSTEM WILL PRINT "EXECUTION COMPLETED".

TAPE

PURPOSE: THE TAPE COMMAND IS USED TO INPUT A PROGRAM WHICH HAS BEEN PUNCHED ON A PAPER TAPE EITHER BY HAND OR THE TPUNCH COMMAND. CONTROL A'S (^A) ARE ACCEPTABLE LEADER AND TRAILER.

INPUT: AFTER THE MONITOR HAS RESPONDED WITH A PERIOD, THE USER TYPES 'TAPE'. AT THIS TIME, THE TAPE SHOULD BEGIN TO FEED AUTOMATICALLY. IT MUST BE NOTED THAT THE TAPE MODE MAY BE USED ONLY ON TERMINALS EQUIPPED WITH THE X-ON AND X-OFF FEATURE, AS EACH LINE MUST BE FOLLOWED BY X-OFF RETURN LINEFEED.

LOAD

PURPOSE: THIS COMMAND ENABLES THE USER TO LOAD HIS PROGRAM FROM THE TERMINAL WITHOUT USING A PAPER TAPE.

INPUT: AFTER THE SYSTEM HAS TYPED ITS PERIOD, THE USER TYPES 'LOAD'. THE SYSTEM RESPONDS WITH "PROGRAM NAME--". THEN THE USER TYPES IN HIS PROGRAM NAME, WHICH IS LIMITED TO TEN PRINTING CHARACTERS. THE PROGRAM IS THEN TYPED IN.

CONTROL Y

PURPOSE: THE CONTROL Y (^Y) COMMAND IS USED TO EXIT FROM THE LOAD AND TAPE MODES.

INPUT: WHEN THE USER HAS FINISHED TYPING IN HIS PROGRAM, HE PRESSES THE 'CONTROL' AND 'Y' KEYS AT THE SAME TIME, AND THEN 'RETURN'. HE IS THEN RETURNED TO MONITOR.

LIST

PURPOSE: THIS COMMAND PRODUCES A LISTING OF THE CURRENT FORTRAN PROGRAM AND DATA.

INPUT: THE USER SIMPLY TYPES 'LIST'.

TPUNCH

PURPOSE: THIS COMMAND PUNCHES A PROGRAM ONTO PAPER TAPE AND SUBSTITUTES THE TAB CHARACTER, CONTROL I (^I), FOR SPACES IN THE LINE NUMBER FIELD. A CONTROL Y (^Y) IS PUNCHED AT THE END OF THE TAPE SO THAT THE TAPE IS READY FOR LOADING.

INPUT: AFTER THE PERIOD HAS BEEN PRINTED BY THE SYSTEM, THE USER TYPES 'TPUNCH', AND TURNS THE TAPE PUNCH ON.

UJOB

PURPOSE: THIS COMMAND ENABLES THE USER TO EXIT THE FORTRAN PACKAGE WITHOUT ERASING THE CURRENT PROGRAM.

INPUT: UPON RECIEPT OF A PERIOD, THE USER TYPES 'UJOB'. THE SYSTEM RESPONDS WITH A LOGOUT MESSAGE, AND THEN RETURNS THE USER TO BASIC.

KJOB

PURPOSE: THIS COMMAND IS USED TO EXIT THE PACKAGE AND ERASE THE CURRENT FORTRAN PROGRAM.

INPUT: THE USER TYPES 'KJOB' AFTER THE SYSTEM HAS GIVEN HIM THE PERIOD.

HELP

PURPOSE: THE HELP COMMAND IS DESIGNED TO GIVE SUMMARIES OF USEFUL INFORMATION IN REGARD TO THE OPERATION OF OMSI 2000 FORTRAN. A SAMPLE OPERATION TAPE IS AVAILABLE FOR USE ON TERMINALS EQUIPPED WITH THE X-ON, X-OFF FEATURE.

INPUT: IF THE USER NEEDS AID, HE SHOULD WAIT FOR A PERIOD, AND THEN TYPE 'HELP'.

THE COMMANDS ON THIS PAGE DEAL WITH OPERATING A LIBRARY OF FORTRAN PROGRAMS. TO DO SO IN THE 2000B/E VERSION OF OMSI 2000 FORTRAN, THE FILE FTNFLB MUST BE AVAILABLE ON A READ/WRITE BASIS. IF IT IS NOT AVAILABLE, THE SYSTEM WILL PRINT EITHER "MISSING OR PROTECTED FILE" OR "WRITE TRIED ON READ ONLY FILE". MISSING OR PROTECTED FILE MEANS THAT THE FILE FTNFLB MUST BE OPENED BEFORE ANY LIBRARY COMMANDS CAN BE EXECUTED. WRITE TRIED ON READ ONLY FILE MEANS THAT SOMEONE ELSE IS USING THE FILE, AND UNLESS THE USER IS IN AN AXXX ACCOUNT, HE WILL NOT BE ABLE TO STORE OR ERASE A PROGRAM. IN EITHER CASE, TO REENTER THE FORTRAN MONITOR, THE USER MUST TYPE 'GET-FORTRN' AND 'RUN'.

USERS OF THE 2000C/F VERSION SHOULD READ THE "RA" SECTION ON PAGE 2-1 FOR DIFFERENCES BEFORE ATTEMPTING TO USE THESE COMMANDS.

STORE

PURPOSE: THE STORE COMMAND IS USED TO STORE THE CURRENT FORTRAN PROGRAM IN THE FILE FTNFLB.

INPUT: THE USER TYPES STORE.

RETRIEVE

PURPOSE: THE PURPOSE OF THE RETRIEVE COMMAND IS TO ENABLE THE USER TO RETRIEVE A PROGRAM WHICH HAS BEEN PREVIOUSLY STORED.

INPUT: THE USER TYPES AN R, THEN A BLANK SPACE, THEN THE NAME OF THE PROGRAM WHICH IS TO BE RETRIEVED.

ERASE

PURPOSE: THIS COMMAND IS USED TO ERASE A PROGRAM WHICH HAS BEEN PREVIOUSLY STORED.

INPUT: THE USER TYPES AN E, FOLLOWED BY A BLANK SPACE. HE THEN TYPES THE NAME OF THE PROGRAM WHICH HE WISHES TO HAVE DELETED.

CATALOG

PURPOSE: THE CATALOG COMMAND GIVES THE USER A LIST OF THE NAMES AND LENGTHS OF PROGRAMS WHICH HAVE BEEN STORED.

INPUT: THE USER TYPES CATALOG.

ASSIGNMENT STATEMENT

THE ASSIGNMENT STATEMENT WILL SET THE VARIABLE FOUND ON THE LEFT OF THE EQUALS SIGN EQUAL TO THE VALUE OF THE EXPRESSION FOUND ON THE RIGHT. THE EXPRESSION ON THE RIGHT IS LIMITED TO TWENTY-FIVE ELEMENTS; EACH CONSTANT, VARIABLE, OPERATOR AND PARENTHESIS COUNT AS ONE ELEMENT.

IN THE FORTRAN PACKAGE, THERE ARE TWO MODES FOR ARITHMETIC CALCULATIONS WHICH ARE: REAL AND INTEGER. IF MORE THAN ONE MODE APPEARS IN A STATEMENT, IT MUST BE COVERED BY THE APPROPRIATE MODE-ALTERING FUNCTION (FUNCTIONS ARE DESCRIBED ON PAGE 1-11).

FORTRAN WILL AUTOMATICALLY CORRECT THE MODE OF THE VALUE OF THE EXPRESSION TO MATCH THE MODE OF THE VARIABLE ON THE LEFT OF THE EQUALS SIGN.

IN A SINGLE EXPRESSION, THE PRIORITY OF OPERATION GOES AS FOLLOWS: FIRST EXPONENTIATION, THEN MULTIPLICATION AND DIVISION, THEN ADDITION AND SUBTRACTION. WITHIN THESE GROUPS, OPERATIONS ARE DONE FROM LEFT TO RIGHT. QUANTITIES WITHIN PARENTHESES WILL BE COMPUTED FIRST.

EXAMPLE:

```
200   KEGG=KEGG+87
      ALPHA=SQRT(34.85)
9     JOGG=INT(SQRT(FLOAT(JOGGER)))
      AREA=3.14159*RADIUS**2.0
```

AVAILABLE OPERATORS ARE:

+	ADDITION	-	SUBTRACTION
*	MULTIPLICATION	/	DIVISION
**	EXPONENTIATION		

READ

PURPOSE: THE READ STATEMENT IS USED TO ENTER VALUES TO THE EXECUTING PROGRAM. IT MUST USE A FORMAT AND THE DEVICES ARE (1) INPUT FROM THE TERMINAL, AND (2) READ FROM A DATA LINE. THE VARIABLES TO WHICH THE DATA IS BEING TRANSFERRED MUST BE LISTED AFTER THE FORMAT REFERENCE. PROGRAM DATA MUST COME IMMEDIATELY AFTER THE END STATEMENT, AND MUST MATCH THE FORMAT WITH WHICH IT IS TO BE READ. WHEN THE SIMULATOR IS READY TO ACCEPT INPUT FROM THE TERMINAL, IT WILL PRINT AN ASTERISK. TYPING A CONTROL B (^B) IN RESPONSE TO THE ASTERISK WILL RETURN THE USER TO THE MONITOR.

EXAMPLE:

```
60    READ(1,100)A,BETA,QUIRP
      READ(2,250)A,B,C
```

WRITE

PURPOSE: THE WRITE STATEMENT IS USED TO TRANSFER VALUES TO THE USER'S TERMINAL. ALL WRITE STATEMENTS MUST REFERENCE THE TERMINAL, THE ONLY OUTPUT DEVICE (3), AND MUST ALSO BE USED IN CONJUNCTION WITH A FORMAT STATEMENT.

EXAMPLE:

```
70    WRITE(3,100)IMP,KZAP,ALPHA
      WRITE(3,265)GAMMA
500   WRITE(3,200)
```

GO TO

PURPOSE: THE GO TO STATEMENT IS USED TO TRANSFER CONTROL TO ANOTHER PART OF THE PROGRAM. IF N IS THE STATEMENT NUMBER OF AN EXECUTABLE STATEMENT, AND A GO TO N STATEMENT IS EXECUTED, CONTROL WILL BE TRANSFERED TO STATEMENT N.

EXAMPLE:

```
      GO TO 750
```

DO

PURPOSE: THE DO STATEMENT CAUSES A PREDETERMINED SET OF INSTRUCTIONS TO BE REPEATED A DESIGNATED NUMBER OF TIMES BY INCREMENTING AN INTEGER VARIABLE BETWEEN TWO ASSIGNED LIMITS. IT IS SIMILAR TO THE FOR STATEMENT IN BASIC CAUSING THE STATEMENTS BEGINNING WITH THE STATEMENT IMMEDIATELY AFTER THE DO STATEMENT AND ENDING WITH THE DESIGNATED STATEMENT TO BE REPEATED. A CONTINUE STATEMENT IS PROVIDED TO TERMINATE DO LOOPS; HOWEVER, ANY STATEMENT EXCEPT GO TO, IF, OR DO MAY BE USED. THE INTITAL VALUE, FINAL VALUE, AND STEP SIZE, IF PRESENT, MAY BE EITHER AN INTEGER CONSTANT OR AN INTEGER VARIABLE. NEGATIVE STEP SIZES ARE ACCEPTABLE. THE INDEX VARIABLE IS UNDEFINED AFTER THE DO IS COMPLETED, AND AS A RESULT, IT IS UNADVISABLE TO JUMP INTO OR OUT OF THE RANGE OF A DO LOOP, AS DOING SO MIGHT HAVE UNEXPECTED RESULTS. SEE SECTION 4, "DEMONSTRATION PROGRAMS" FOR SAMPLES OF DO LOOPS.

EXAMPLE:

```
      DO 90 INK=1,IMP
275   DO 150 I=2,10,2
```

CONTINUE

PURPOSE: THE CONTINUE STATEMENT IS A DUMMY STATEMENT AND MAY BE USED ANYWHERE IN A PROGRAM. THIS STATEMENT IS OFTEN USED AS THE TERMINAL STATEMENT OF A DO LOOP.

EXAMPLE:

```
595   CONTINUE
```

FORMAT

PURPOSE: THE FORMAT STATEMENT IS USED TO GIVE THE SPECIFICATIONS BY WHICH INPUT AND OUTPUT ARE FORMATTED. FORMAT IS A NON-EXECUTABLE STATEMENT, AND ALL NON-EXECUTABLE STATEMENTS MUST PRECEED ANY EXECUTABLE STATEMENT. IN THE FOLLOWING SECTION, 'N' MEANS AN INTEGER REPETITION FACTOR FROM 1 TO 72, INCLUSIVE, 'W' STANDS FOR THE TOTAL FIELD WIDTH, THE MAXIMUM SET AT 20, AND 'D' REPRESENTS THE NUMBER OF DIGITS WHICH ARE TO GO TO THE RIGHT OF THE DECIMAL POINT, IF PRESENT. BOTH 'W' AND 'D' MUST BE INTEGERS.

SPECIFICATION	FUNCTION
NIW	USED TO TRANSMIT N INTEGERS WHICH ARE W DIGITS IN LENGTH (BLANKS ARE TREATED AS ZEROES)
NFW.D	USED TO TRANSMIT N FLOATING POINT VALUES WHICH HAVE A TOTAL WIDTH OF W (INCLUDING THE DECIMAL POINT) AND D DIGITS TO THE RIGHT OF THE DECIMAL POINT.
NEW.D	USED TO TRANSMIT N FLOATING POINT VALUES IN EXPONENTIAL FORMAT. SIMILAR TO F, BUT THE FIELD WIDTH MUST ALSO INCLUDE THE EXPONENT.
NX	BLANK FIELD N SPACES WIDE.
'TEXT'	USED TO TRANSMIT LITERALS TO THE TELETYPE.

GROUPINGS WITHIN FORMAT SPECIFICATIONS MUST BE SEPARATED BY EITHER A COMMA OR A SLASH. A SLASH USED TO SEPARATE FORMAT SPECIFICATIONS WILL GENERATE A RETURN LINEFEED, AND THEN GO TO THE NEXT GROUPING. IF MORE THAN ONE SLASH APPEARS IN A ROW (AS IN THE EXAMPLE NUMBERED 670), AS MANY RETURN-LINEFEEDS WILL BE PRINTED AS THERE ARE SLASHES.

IN ANY SPECIFICATION WHERE N IS PRESENT, IT MAY BE OMITTED, AS THE COMPILER WILL ASSUME 1 IF IT IS NOT PRESENT.

EXAMPLE:

```
100   FORMAT(F6.2,F7.3)
200   FORMAT(E10.3,I5,I8)
240   FORMAT('THE ANSWER IS ',F9.4)
75    FORMAT(8X,'YOU HAVE ',F7.2,'COWS'/15X,F7.2,'PIGS'//)
670   FORMAT(10X,E3.9//10X,F7.3,4I8,2F3.0)
```

STOP

PURPOSE: THE STOP STATEMENT IS USED TO STOP THE EXECUTION OF A PROGRAM. IF AN INTEGER IS PLACED AFTER THE WORD STOP, IT WILL BE PRINTED WHEN THE STOP IS EXECUTED.

EXAMPLE:

```
      STOP
500   STOP 350
```

IF

PURPOSE: THE IF STATEMENT IS USED TO PROVIDE A CONDITIONAL TRANSFER. THE EXPRESSION IN THE IF STATEMENT IS EVALUATED, AND CONTROL IS PASSED TO THE FIRST STATEMENT REFERENCE IF THE VALUE OF THE EXPRESSION IS NEGATIVE, THE MIDDLE IF ZERO, AND THE THIRD IF POSITIVE.

EXAMPLE:

```
      IF (X-3.6) 200,350,350
70    IF (INK*IMP) 500,510,500
```

END

PURPOSE: THE END STATEMENT SIGNALS THE END OF THE PROGRAM TO THE COMPILER. ANYTHING AFTER THE END IS ASSUMED TO BE DATA, THUS IGNORED.

EXAMPLE:

```
      END
```

COMMENT

PURPOSE: THE C STATEMENT IS USED TO INSERT NON-EXECUTABLE COMMENTS INTO THE LISTING OF A PROGRAM. C STATEMENTS ARE NOT COMPILED OR EXECUTED. THE C MUST OCCUR IN THE FIRST COLUMN.

EXAMPLE:

```
C  THIS IS AN EXAMPLE OF A COMMENT
```

FUNCTIONS

ON THIS PAGE, IX MEANS AN INTEGER CONSTANT, VARIABLE OR EXPRESSION, AND RX MEANS A REAL CONSTANT, VARIABLE OR EXPRESSION.

FUNCTION	TYPE OF ARGUMENT	TYPE OF VALUE	VALUE RETURNED
SQRT SQRT(RX)	REAL	REAL	SQUARE ROOT OF A POSITIVE ARGUMENT.
ABS ABS(RX)	REAL	REAL	ABSOLUTE VALUE OF THE ARGUMENT.
IABS IABS(IX)	INTEGER	INTEGER	ABSOLUTE VALUE OF THE ARGUMENT.
AINT AINT(RX)	REAL	REAL	LARGEST INTEGER LESS THAN OR EQUAL TO THE VALUE OF THE ARGUMENT. (TRUNCATION)
INT INT(RX)	REAL	INTEGER	LARGEST INTEGER LESS THAN OR EQUAL TO THE VALUE OF THE ARGUMENT. (TRUNCATION)
ATAN ATAN(RX)	REAL	REAL	ARCTANGENT OF THE ARGUMENT EXPRESSED IN RADIANS.
COS COS(RX)	REAL	REAL	COSINE OF THE ARGUMENT EXPRESSED IN RADIANS.
SIN SIN(RX)	REAL	REAL	SINE OF THE ARGUMENT EXPRESSED IN RADIANS.
EXP EXP(RX)	REAL	REAL	VALUE OF "E" TO THE POWER OF THE ARGUMENT.
ALOG ALOG(RX)	REAL	REAL	NATURAL LOGARITHM OF THE ARGUMENT.
ALOG10 ALOG10(RX)	REAL	REAL	COMMON LOGARITHM OF THE ARGUMENT.
FLOAT FLOAT(IX)	INTEGER	REAL	CONVERTS AN INTEGER ARGUMENT TO A REAL VALUE.
IFIX IFIX(RX)	REAL	INTEGER	TRUNCATES THE REAL ARGUMENT AND CONVERTS IT TO AN INTEGER VALUE.

EDITING

IN OMSI 2000 FORTRAN EDITING IS DONE DIRECTLY THROUGH THE MONITOR. ENTIRE LINES ARE CHANGED, USING COMMANDS DESCRIBED BELOW. USE OF THE LN COMMAND IS RECOMMENDED BEFORE ATTEMPTING TO EDIT, AS DOING SO WILL MAKE EDITING MUCH EASIER AND MORE ACCURATE.

COMMAND	PURPOSE
NI-STRING	THIS WILL INSERT THE STRING BEFORE THE NTH LINE IN THE PROGRAM.
ND	THIS COMMAND DELETES THE NTH LINE.
NC-STRING	THIS COMMAND CHANGES THE NTH LINE TO THE STRING.
A-STRING	THIS WILL APPEND THE STRING TO THE END OF THE PROGRAM.
LN	THIS COMMAND WILL PRODUCE A LISTING OF THE CURRENT PROGRAM WITH SEQUENCING LINE NUMBERS (ENCLOSED WITHIN PARENTHESES) PLACED BEFORE EACH LINE. IT MUST BE NOTED THAT THESE NUMBERS ARE FOR CONVENIENCE ONLY, AND ARE NOT A PART OF THE PROGRAM.

IT SHOULD BE NOTED THAT THE I AND D COMMANDS WILL EFFECT THE SEQUENCING LINE NUMBERS OBTAINED BY LN.

EXAMPLE:

```
.LI
SAMPLE
10    FORMAT(F10.4)
      TOTAL=2.0+2.0
      WRITE(3,100 TOTAL
      STOP
      END
.LN
(  1)  SAMPLE
(  2)  10    FORMAT(F10.4)
(  3)        TOTAL=2.0+2.0
(  4)        WRITE(3,100 TOTAL
(  5)        STOP
(  6)        END
.4C-^IWRITE(3,10) TOTAL
.LN
(  1)  SAMPLE
(  2)  10    FORMAT(F10.4)
(  3)        TOTAL=2.0+2.0
(  4)        WRITE(3,10) TOTAL
(  5)        STOP
(  6)        END
.
```

ADDITIONAL FEATURES AVAILABLE IN THE 2000C/F VERSION OF FORTRAN

COMMANDS

LN-M, N

DELIMITERS HAVE BEEN ADDED TO THE LN COMMAND. IF PRESENT, THEY REPRESENT STARTING AND STOPPING POINTS, M AND N RESPECTIVELY, FOR THE LISTING. IF EITHER OR BOTH ARE GONE, THE SYSTEM ASSUMES 1 (ONE) AND THE NUMBER OF LINES IN THE PROGRAM.

O FILE NAME, LENGTH

THE OPEN COMMAND IS USED TO CREATE A FORTRAN DATA FILE IN A BASIC PROGRAM STORAGE FILE. A FILE WHICH HAS BEEN CREATED BY THIS METHOD CAN BE USED IN A FORTRAN PROGRAM TO STORE DATA. MAXIMUM FILE SIZE IS 100 ELEMENTS.

RA

IT IS POSSIBLE TO USE MORE THAN ONE FILE TO STORE FORTRAN PROGRAMS. WHEN THE FIRST PROGRAM STORAGE COMMAND (STORE, ERASE, RETRIEVE, OPEN, CATALOG) IS EXECUTED, THE SYSTEM WILL REQUEST A "PROGRAM STORAGE FILE NAME--". THE USER THEN TYPES IN THE NAME OF THE PROGRAM STORAGE FILE. THE FILE NAME IS KEPT SO THAT ALL SUBSEQUENT PROGRAM STORAGE COMMANDS AT THAT SITTING WILL BE EXECUTED WITHOUT DELAY. IF THE USER WISHES TO SWITCH TO ANOTHER FILE, HE MAY DO SO BY TYPING 'RA', WHICH STANDS FOR REASSIGN, AND THE SYSTEM WILL REQUEST ANOTHER PROGRAM STORAGE FILE NAME.

WHEN FORTRAN IS FIRST RUN, THE STATUS OF THE FILES IS CHECKED AND REPORTED TO THE USER IF ANY OF THE SYSTEM FILES ARE NOT AVAILABLE. ALSO, THE STATUS OF THE PROGRAM STORAGE FILE IS CHECKED; IF IT IS READ ONLY, THE SYSTEM WILL ACCEPT CATALOG, AND RETRIEVE COMMANDS, BUT WILL NOT ACCEPT STORE, ERASE, OR OPEN COMMANDS.

PUNCH

THE PUNCH COMMAND IS USED TO CREATE A PAPER TAPE OF THE CURRENT FORTRAN PROGRAM. BLANKS IN THE STATEMENT NUMBER FIELD WILL NOT BE REPLACED BY THE TAB CHARACTER. ESSENTIALLY, A LIST IS PERFORMED WITH NULL LEADER AND TRAILER. TO CREATE A TAPE IN THIS MANNER, THE USER TYPES PUNCH.

STATEMENTS

REAL AND INTEGER

PURPOSE: THE PURPOSE OF THE REAL AND INTEGER STATEMENTS IS TO PROVIDE A MEANS TO OVERRIDE THE TYPE DEFINITION OF VARIABLES. FOR EXAMPLE, TO HAVE 'ALPHA' DECLARED TO BE AN INTEGER VARIABLE, AND 'IMP' DECLARED TO BE A REAL VARIABLE, THE USER MIGHT DO THE FOLLOWING:

EXAMPLE:

```
        REAL IMP,IMP1
        INTEGER ALPHA
```

REAL AND INTEGER STATEMENTS ARE NON-EXECUTABLE.

LOGICAL IF

PURPOSE: THE PURPOSE OF THE LOGICAL IF STATEMENT IS TO PROVIDE A MEANS TO EXECUTE A STATEMENT ONLY IF CERTAN CONDITIONS ARE TRUE. SEVERAL NEW OPERATORS HAVE BEEN ADDED TO MAKE THIS STATEMENT MORE USEFUL. THEY ARE:

.AND. .OR. .EOR. (EXCLUSIVE OR)

.EQ. (EQUAL TO) .NE. (NOT EQUAL TO)
.LT. (LESS THAN) .LE. (LESS THAN OR EQUAL TO)
.GT. (GREATER THAN) .GE. (GREATER THAN OR EQUAL TO)

EACH OF THESE OPERATORS RETURNS A ONE IF THE EXPRESSION IS TRUE; A ZERO IF IT IS FALSE. FOR EXAMPLE: IF IMP HAS THE VALUE 1, THEN (IMP.EQ.1) WOULD BE ONE, AND (IMP.GE.7) WOULD HAVE A VALUE OF ZERO. THE FORM OF THE LOGICAL IF STATEMENT IS THIS:

```
IF(<EXPRESSION>) <STATEMENT>
```

WHERE <EXPRESSION> IS ANY LEGAL EXPRESSION, AND <STATEMENT> IS ANY EXECUTABLE FORTRAN STATEMENT EXCEPT DO, IF OR END. IF <EXPRESSION> IS TRUE (NON-ZERO), THEN THE <STATEMENT> WILL BE EXECUTED.

EXAMPLE:

```
100     IF(OOPS.EQ.3) WRITE(3,105) OOPNO
        IF(ALPHA.LT.0.0) GOTO 150
        IF(GANGRE.NE.5.5) READ(2,10) GANGRE
        IF(FAST) FAST=FAST+1.0
```

ADDITIONAL FUNCTIONS AVAILABLE IN THE 2000C/F VERSION OF FORTRAN

FUNCTION	TYPE AND NUMBER OF ARGUMENTS	TYPE OF VALUE	DESCRIPTION
AMOD AMOD(R1X, R2X)	2 REAL	REAL	OBTAIN THE REMAINDER. RETURNS R1X-(R1X/R2X)*R2X (MODULAR ARITHMETIC)
MOD MOD(I1X, I2X)	2 INTEGER	INTEGER	OBTAIN THE REMAINDER. RETURNS I1X-(I1X/I2X)*I2X (MODULAR ARTIHMETIC)
AMAX1	2 OR MORE REAL	REAL	RETURNS THE LARGEST ALGEBRAIC ARGUMENT.
MAX1	2 OR MORE REAL	INTEGER	
AMAX0	2 OR MORE INTEGER	REAL	
MAX0	2 OR MORE INTEGER	INTEGER	
AMAX1(R1X, R2X, . . . , RNX)			
AMIN1	2 OR MORE REAL	REAL	RETURNS THE SMALLEST ALGEBRAIC ARGUMENT.
MIN1	2 OR MORE REAL	INTEGER	
AMIN0	2 OR MORE INTEGER	REAL	
MIN0	2 OR MORE INTEGER	INTEGER	
AMIN1(R1X, R2X, . . . , RNX)			
SIGN	2 REAL	REAL	RETURNS THE ABSOLUTE VALUE OF THE FIRST TIMES THE SIGN OF THE SECOND.
ISIGN	2 INTEGER	INTEGER	
SIGN(R1X, R2X)			
DIM	2 REAL	REAL	RETURNS THE POSITIVE DIFFERENCE BETWEEN THE TWO ARGUMENTS.
IDIM	2 INTEGER	INTEGER	
DIM(R1X, R2X)			

PAUSE

PURPOSE: THE PAUSE STATEMENT CAUSES PROGRAM EXECUTION TO COME TO A HALT FOR A PERIOD OF FIVE SECONDS. IF THE USER TYPES A CONTROL B (^B) DURING THE EXECUTION OF A PAUSE STATEMENT, EXECUTION WILL BE TERMINATED. IF AN INTEGER IS PLACED AFTER THE WORD PAUSE, IT WILL BE PRINTED UPON EXECUTION.

EXAMPLE:

```
100   PAUSE
      PAUSE 472
```

DATA

PURPOSE: THE DATA INTIALIZATION STATEMENT IS A NON-EXECUTABLE STATEMENT WHOSE PURPOSE IS TO ASSIGN INITIAL VALUES TO NON-SUBSCRIPTED VARIABLES.

EXAMPLE:

```
        DATA I,J,FLAG1/0,34,0.0/
```

WOULD SET I=0, J=34, FLAG1=0.0

```
        DATA IMP,IMP1/10,20/,IFLAG,NVALUE/0,5E+03/
```

WOULD SET IMP=10, IMP1=20, IFLAG=0, NVALUE=5000

FORMAT

H FORMATTING IS AVAILABLE. ITS FORM IS NHTEXT, WHERE N REPRESENTS THE NUMBER OF CHARACTERS AFTER THE H WHICH ARE INCLUDED IN THE LITERAL FIELD. ON OUTPUT, THE LITERAL FIELD AFTER THE H IS PRINTED. ON INPUT, ESSENTIALLY AN NX IS PERFORMED.

CARRIAGE CONTROL HAS BEEN ADDED TO FORMAT STATEMENTS. THE FIRST CHARACTER OF THE FIRST HOLLERITH FIELD OR LITERAL STRING IS THE CARRIAGE CONTROL CHARACTER.

CHARACTER	CARRIAGE RESPONSE
1	FORMFEED (10 LINEFEEDS)
0	DOUBLE SPACING
#	NO CARRIAGE RETURN OR LINEFEED
+	CARRIAGE RETURN ONLY
-	LINEFEED ONLY

ANY OTHER CHARACTER WILL CAUSE SINGLE SPACING. THE FORMFEED WILL BE PRINTED BEFORE THE REST OF THE SPECIFICATIONS; IN ALL OTHER CASES, THE CARRIAGE CONTROL WILL BE DONE AT THE END OF THE WRITE.

EXAMPLE:

```
100   FORMAT(33H1THIS IS A SAMPLE HOLLERTIH FIELD)
255   FORMAT(8H0ERROR**,2I3,10H RUN ENDED)
90    FORMAT(9H YOU HAVE,F7.2,7H COWS,  ,F7.2,8H HORSES.)
```

COMPUTED GOTO

PURPOSE: THE COMPUTED GOTO STATEMENT PROVIDES FOR CONDITIONAL TRANSFER TO ONE OF SEVERAL STATEMENTS IN THE PROGRAM. FROM A LIST OF STATEMENT NUMBERS, ONE IS CHOSEN DEPENDING UPON THE VALUE OF A NON-SUBSCRIPTED INTEGER VARIABLE.

EXAMPLE:

```
GOTO (7,24,56,34),IMP
GOTO (7,24,56,34)IMP
```

WHEN THIS STATEMENT IS EXECUTED, CONTROL WOULD BE PASSED TO STATEMENT 7 IF IMP WAS 1; STATEMENT 24 IF IMP WAS 2; STATEMENT 56 IF IMP WAS 3 OR STATEMENT 34 IF IMP WAS 4. IF THE VALUE OF IMP WAS LESS THEN ONE OR GREATER THAN THE NUMBER OF STATEMENT NUMBERS LISTED, AN ERROR MESSAGE WOULD RESULT.

SUBSCRIPTED VARIABLES

SUBSCRIPTED VARIABLES ARE AVILABLE. ONE, TWO, OR THREE DIMENSIONAL ARRAYS ARE ACCEPTABLE. SUBSCRIPTED VARIABLE NAMES MUST CONFORM TO ALL RULES SET FOR VARIABLE NAMES, AND MUST BE DIMENSIONED IN A DIMENSION STATEMENT. THE DIMENSION STATEMENT MUST BE THE FIRST STATEMENT IN THE PROGRAM. IF AN ARRAY NAME APPEARS IN A TYPE STATEMENT (REAL OR INTEGER), THE MODE OF THE ENTIRE ARRAY WILL BE CHANGED. THE TOTAL NUMBER OF SUBSCRIPTED VARIABLES MUST NOT EXCEED 200 IN AT MOST 12 SEPARATE VARIABLES.

EXAMPLE:

```
DIMENSION ALPHA(3,10),ZAPPA(4,6,2),J(15)
```

SUBSCRIPTED VARIABLES MAY BE USED IN ASSIGNMENT, READ, WRITE, AND IF STATEMENTS. SUBSCRIPTS MAY BE AN INTEGER CONSTANT OR AN INTEGER VARIABLE.

EXAMPLE:

```
      DIMENSION IMP(10,10)
100   FORMAT(I3)
200   FORMAT('#',I4)
300   FORMAT(/)
      DO 301 INK1=1,10
      DO 301 INK2=1,10
301   READ(1,100)IMP(INK1,INK2)
      DO 303 INK1=1,10
      DO 302 INK2=1,10
      IMP(INK1,INK2)=IMP(INK1,INK2)+INK1*INK2
302   WRITE(3,200)IMP(INK1,INK2)
303   WRITE(3,300)
      STOP
      END
```

DATA FILES

A DATA FILE IS A PERMANENT DATA STORAGE AREA FOR NUMERIC VALUES. MAXIMUM FILE SIZE IS 100 ELEMENTS, AND ONLY ONE FILE MAY BE USED AT A TIME. DATA WRITTEN ONTO A FILE WILL BE PHYSICALLY WRITTEN ONTO STORAGE AFTER EXECUTION OF A STOP STATEMENT, AND AS A RESULT, DATA WRITTEN ONTO A FILE IS LOST WHEN EXECUTION IS ABORTED, EITHER BY AN ERROR MESSAGE, TYPING CONTROL B (^B) IN RESPONSE TO A PAUSE OR READ, OR PRESSING 'BREAK'.

THE STATEMENTS LISTED BELOW ARE USED TO SPECIFY WHICH FILE IS TO BE USED, WHAT VALUES ARE TO BE WRITTEN ONTO AND READ FROM THE FILE, AND TO CONTROL THE FILE POINTER.

DEFINE FILE

PURPOSE: THE DEFINE FILE STATEMENT IS USED TO SPECIFY WHICH FORTRAN DATA FILE IN WHICH BASIC PROGRAM STORAGE FILE IS TO BE RETRIEVED. IT IS A NON-EXECUTABLE STATEMENT.

EXAMPLE:

```
DEFINE FILE(BASICF,FORTRNDATA)
```

CALLS THE ENTRY "FORTRNDATA" FROM THE BASIC FILE "BASICF"

```
DEFINE FILE(JELLO,DATAFILE1)
```

CALLS THE ENTRY "DATAFILE1" FROM THE BASIC FILE "JELLO"

FILE WRITE

PURPOSE: THE FILE WRITE STATEMENT IS USED TO WRITE THE VALUE OF VARIABLES ONTO THE DATA FILE. IT HAS THE FORM:

```
WRITE(5) VAR,VAR, ...
```

WHERE THE DEVICE CODE "5" INDICATES A WRITE ONTO THE FILE. THERE MUST NOT

FILE CONTAINS UNFORMATTED, UNTYPED DATA. THE VARIABLES ARE WRITTEN OONTO THE FILE, AND THE FILE POINTER IS ADVANCED TO THE LOCATION IMMEDIATELY FOLLOWING THE LAST VALUE WRITTEN ON THE FILE.

EXAMPLE:

```
105    WRITE(5) INK,IMP,ALPHA
       WRITE(5) I1,I2,I3,ARRAY(I1,I2,I3)
```

FILE READ

PURPOSE: THE PURPOSE OF THE FILE READ STATEMENT IS TO ALLOW VALUES WHICH HAVE BEEN WRITTEN ON A FILE TO BE RETRIEVED. IT HAS THE FORM:

```
READ(4) VAR,VAR, ...
```

WHERE THE DEVICE CODE "4" INDICATES A READ FROM THE FILE. THERE MUST NOT
READING VALUES FROM THE FILE, THE TYPE OF THE VALUE IS ADJUSTED (AND TRUNCATED, IF NECESSARY) TO MATCH THE TYPE OF THE VARIABLE INTO WHICH IT IS BEING READ. THE FILE POINTER IS POSITIONED IMMEDIATELY AFTER THE LAST VALUE WHICH WAS READ.

EXAMPLE:

```
100    READ(4) IMP, INK, AL(ISUB)
       READ(4) IBOBO, IMP
```

REWIND

PURPOSE: THE REWIND STATEMENT ALLOWS THE DATA IN A FILE TO BE RE-USED BY RESTORING THE FILE POINTER TO THE BEGINNING OF THE FILE.

EXAMPLE:

```
       REWIND
```

BACKSPACE

PURPOSE: THE BACKSPACE STATEMENT IS USED TO BACKSPACE THE FILE POINTER ONE POSITION TO ALLOW DATA ITEMS TO BE SELECTIVELY RE-READ.

EXAMPLE:

```
       BACKSPACE
```

FORTRAN FILE STRUCTURE

TO RUN THE BASIC PACKAGE, THERE MUST BE THREE FILES AVAILABLE. THESE ARE: FTNFLS, FTNULT, AND FTNCOP. THESE FILES MUST BE PRESENT TO RUN ANY VERSION OF FORTRAN. FTNFLS HOLDS THE LISTING OF THE FORTRAN PROGRAM. WHEN THE PROGRAM IS EXECUTED, IT IS COMPILED INTO THE FILE FTNCOP, WHILE FTNULT HOLDS PROGRAM DATA AND OTHER INFORMATION THE COMPILER MUST USE. ANY EDITING COMMANDS USE THE FILES FTNFLS AND FTNULT.

THE RECOMMENDED FILE SIZES TO RUN A 25 STEP PROGRAM ARE:

2000B/E		2000C/F	
FTNULT	8	FTNULT	6
FTNCOP	25	FTNCOP	25
FTNFLS	10	FTNFLS	6

TO RUN A 100 STEP PROGRAM, THESE FILE SIZES MIGHT BE USED:

2000B/E		2000C/F	
—		FTNULT	8
—		FTNCOP	100
—		FTNFLS	25

IF THE SYSTEM PRINTS "PROGRAM IS TOO LONG" AFTER THE USER HAS TYPED THE LOAD COMMAND, THE FILE FTNFLS IS TOO SMALL. IF THE SYSTEM RESPONDS WITH THE "PROGRAM IS TOO LONG" MESSAGE AFTER THE USER HAS TYPED THE EXECUTE COMMAND, THE FILE FTNCOP IS TOO SMALL. IN EITHER CASE, THE FILE MUST EITHER BE ENLARGED OR THE PROGRAM MUST BE SHORTENED FOR THE SYSTEM TO LOAD AND EXECUTE THE PROGRAM.

IN THE 2000B/E VERSION, IF THE USER WISHES TO OPERATE A LIBRARY OF STORED PROGRAMS, HE MUST OPEN A BASIC FILE NAMED FTNFLB. AS THIS FILE WILL CONTAIN LISTINGS OF STORED PROGRAMS, IT IS ADVISABLE TO OPEN THE FILE FOR AT LEAST 10 RECORDS.

USERS OF THE 2000C/F VERSION MAY OPEN FILES WITH ANY NAMES, AND RESPOND WITH THE NAME OF THE FILE THEY WISH TO USE WHEN A "PROGRAM STORAGE FILE NAME--" IS REQUESTED.

AS IN THE 2000B/E VERSION, THE PROGRAM STORAGE FILE WILL HOLD LISTINGS OF THE STORED PROGRAMS, THEREFORE, A MINIMUM OF 5 RECORDS IS RECOMMENDED. THE PROGRAM STORAGE FILE IS CHECKED BEFORE IT IS USED BY FORTRAN; IF IT IS NOT AN ACCEPTABLE PROGRAM STORAGE FILE WHICH HAS BEEN USED BEFORE BY FORTRAN FOR THAT PURPOSE, IT WILL BE ERASED AND FLAGGED AS BEING A FORTRAN PROGRAM STORAGE FILE.

SPECIAL CHARACTERS

THERE ARE SEVERAL SPECIAL CHARACTERS WHICH FORTRN UTILIZES TO MAKE OPERATION EASIER FOR THE USER. THEY ARE:

^B	IS USED TO EXIT FROM A FORTRAN READ OR PAUSE STATEMENT. IF THE FIRST CHARACTER THE USER TYPES IN IS A CONTROL B (^B), THE PROGRAM WILL BE TERMINATED.
^C	USED TO EXIT FROM AN INPUT LOOP, MAY BE USED TO EXIT FROM THE FORTRAN MONITOR TO ENLARGE OR CREATE A FILE, CHECK ON TIME OR SPACE, ETC. USING ^C TO EXIT RETURNS THE USER TO BASIC.
^N	GENERATES A LINEFEED WHEN PRINTED, AND THEREFORE SHOULD BE AVOIDED IN LISTINGS AND PUNCHES OF FORTRN PROGRAMS.
^O	GENERATES A RETURN WHEN PRINTED, AND THEREFORE SHOULD NOT BE INCLUDED IN LISTS OR PUNCHES, AS SO DOING WOULD CAUSE SEVERE GARBLING IF THE TAPE WERE RUN BACK IN.
^X, ESC	IS USED TO DELETE THE ENTIRE CURRENT LINE OF INPUT. ON THE 2000B AND 2000C, USE ESC, ON THE 2000E AND 2000F, USE ^X. ECHOES AS A BACKSLASH.
^Y	IS USED TO EXIT FROM THE LOAD AND TAPE MODES.
<-	IS USED TO DELETE THE CHARACTER IMMEDIATELY PRECEEDING IT. MORE THAN ONE BACKARROW MAY BE USED IN A ROW; EACH BACKARROW DELETES ONE CHARACTER. FOR EXAMPLE: EDECU←←←←XECUTE IS INTERPRETED AS EXECUTE LU<-IST IS INTERPRETED AS LIST (← ECHOES AS "_" ON SOME TERMINALS.)

2000B/E VERSION TAPE LOADING INSTRUCTIONS

THE NAME THAT EACH PROGRAM MUST HAVE IS WRITTEN ON THE SEPARATE FORTRAN SYSTEM TAPE. THE FOLLOWING SET OF COMMANDS IS SUGGESTED FOR LOADING:

```
SCRATCH
NAME-FORTRN
TAPE
 AT THIS POINT, START THE TAPE THROUGH THE
 READER.  WHEN IT IS FINISHED, TYPE
KEY
SAVE
```

THIS PROCESS MUST BE REPEATED FOR EACH OF THE FOLLOWING PROGRAMS (CHANGING THE NAME COMMAND TO THE APPROPRIATE PROGRAM NAME EACH TIME):

FORTRN
FTNLIB
FTNINS
FTNCPL
FTNCP2
FTNRUN
FTNRN2

IF POSSIBLE, THE USER SHOULD CSAVE FTNCPL, FTNCP2, FTNRUN, AND FTNRN2, AS THEY CHAIN FREQUENTLY.

ALL NECESSARY FILES FOR SYSTEM OPERATION ARE LISTED ON PAGE 3-1 OF THIS MANUAL.

2000B/E SYSTEM LIBRARY USE

TO ENABLE THE FORTRAN SYTEM TO RUN FROM A SYSTEM LIBRARY, THE USER MUST MODIFY THE CHAIN STATEMENTS. TO DO SO, HE MUST GET THE PROGRAM, TYPE IN THE INDICATED LINES, AND THEN KILL AND SAVE (OR CSAVE) THE PROGRAM. FOR FORTRAN TO RUN ON A 2000B, ALL QUOTATION MARKS MUST BE REMOVED FROM THE CHAIN STATEMENTS. SEE PAGE 1-1.

NAME	LINES
FORTRN	320 CHAIN "$FTNLIB" 1000 CHAIN "$FTNINS" 1130 CHAIN "$FTNCPL"
FTNCPL	1060 CHAIN "$FTNCP2" 1140 CHAIN "$FORTRN" 1920 CHAIN "$FTNRUN"
FTNCP2	1020 CHAIN "$FTNCPL"
FTNRUN	480 CHAIN "$FORTRN" 580 CHAIN "$FTNRN2"
FTNRN2	320 CHAIN "$FTNRUN" 360 CHAIN "$FORTRN"
FTNLIB	260 CHAIN "$FORTRN"

2000C/F VERSION TAPE LOADING INSTRUCTIONS

THE NAME THAT EACH PROGRAM MUST HAVE IS WRITTEN ON THE SEPRATE FORTRAN SYSTEM TAPES. THE FOLLOWING SET OF COMMANDS IS SUGGESTED FOR LOADING:

```
SCRATCH
NAME-FORTRN
TAPE
 AT THIS POINT, START THE TAPE THROUGH THE
 READER.  WHEN IT IS FINISHED, TYPE
KEY
CSAVE
```

REPEAT THIS PROCESS FOR EACH OF THE FOLLWING PROGRAMS (CHANGING THE NAME COMMAND TO THE APPROPRIATE PROGRAM NAME EACH TIME):

```
FORTRN
FTNCPL
FTNRUN
FTNINS
```

ALL NECESSARY FILES FOR SYSTEM OPERATION ARE LISTED ON PAGE 3-1 OF THIS MANUAL.

2000C/F SYSTEM/GROUP LIBRARY USE

TO ENABLE THE FORTRAN SYSTEM TO RUN FROM A SYSTEM OR GROUP LIBRARY, THE CHAIN STATEMENTS MUST BE MODIFIED. TO DO THIS, THE USER MUST RETRIEVE THE DESIGNATED PROGRAM, TYPE IN THE LINES LISTED BELOW, AND THEN KILL AND CSAVE (OR SAVE) THE PROGRAM.

```
NAME      LINES (FOR GROUP USE, CHANGE $ TO *.)

FORTRN    150  CHAIN "$FTNINS"
          1220 CHAIN "$FTNINS",340
          1370 CHAIN "$FTNCPL"

FTNCPL    940  CHAIN "$FTNRUN"
          960  CHAIN "$FORTRN",100

FTNRUN    220  CHAIN "$FORTRN",5790
          230  CHAIN "$FORTRN",2530

FTNINS    1690 CHAIN "$FORTRN",100
```

SYNTAX AND PROGRAM CONSTRUCTION ERRORS

CHARACTERS AFTER STATEMENT END

-UNNECESSARY LETTERS ARE APPEARING AFTER THE NORMAL END OF THE STATEMENT. THESE CHARACTERS MUST BE REMOVED FOR THE PROGRAM TO RUN.

DATA TABLE ERROR

-ERROR IN THE SYNTAX OF A DATA STATEMENT. CHECK THE DATA STATEMENT SECTION IN THE C/F SECTION OF THE MANUAL.

DUPLICATE LINE NUMBER

-TWO OR MORE STATEMENTS HAVE THE SAME LINE NUMBER.

EXPRESSION TOO COMPLEX

-THE ARITHMETIC EXPRESSION CONTAINS TOO MANY ELEMENTS; IT MUST BE CHANGED TO TWO OR MORE STATEMENTS.

ILLEGAL EXPONENT

-EXPONENT ON CONSTANT IS 40 OR LARGER. THE CONSTANT MUST BE CHANGED TO HAVE AN EXPONENT LESS THEN 40.

ILLEGAL FORMAT

-THE SYNTAX OF THE STATEMENT IS NOT COMPATIBLE WITH THE SYSTEM. CHECK THE STATEMENT CAREFULLY FOR MISPLACED COMMAS, MISSPELLINGS, OR THE LIKE. IF NO OBVIOUS ERROR IS FOUND, CHECK THE MANUAL. A COMMON CAUSE OF THIS ERROR IS USING A VARIABLE NAME THAT IS ALSO A KEYWORD.

ILLEGAL FORMAT SPECIFICATION

-THE FORMAT STATEMENT CONTAINS AN INCORRECT SPECIFICATION. CAUSES INCLUDE THE USE OF THE WRONG LETTER, ATTEMPTING TO USE A FIELD LARGER THAN 20 IN AN I, E OR F SPECIFICATION, OR NOT ALLOWING SPACE FOR THE NECESSARY DECIMAL POINTS, SIGNS OR EXPONENTS IN AN E OR F CODE. CHECK THE FORMAT SECTIONS IN THE MANUAL.

ILLEGAL LINE NUMBER

-THE LINE NUMBER IS NOT A POSITIVE INTEGER LESS THAN ONE HUNDRED THOUSAND.

ILLEGAL OPERAND

-THE STATEMENT TYPE REQUESTED IS NOT AVAILABLE. CHECK SPELLING, AND THEN CHECK MANUAL.

ILLEGAL OR FAULTY SUBSCRIPTS

-THE SUBSCRIPTS ON A VARIABLE WHICH HAS BEEN DIMENSIONED ARE OF THE WRONG FORM. COMMON CAUSES OF THIS ERROR INCLUDE NON-INTEGRAL SUBSCRIPTS, ATTEMPTING TO USE COMPUTED SUBSCRIPTS AND THE USE OF AN ARRAY NAME AS A NON-SUBSCRIPTED VARIABLE.

ILLEGAL PROGRAM CONSTRUCTION

-THE NON-EXECUTABLE STATEMENTS ARE NOT FIRST IN THE PROGRAM OR THE DIMENSION STATEMENT IS NOT FIRST IN THE PROGRAM. THE STATEMENTS MUST BE REARRANGED.

ILLEGAL REPLICATOR

-IN A FORMAT STATEMENT, THE REPLICATOR ON A SPECIFICATION (FOR EXAMPLE THE 4 ON '4I6' IS A REPLICATOR) IS NOT A POSITIVE INTEGER LESS THAN OR EQUAL TO 72.

ILLEGAL VARIABLE

-A VARIABLE WITH EITHER MORE THAN 6 CHARACTERS OR HAVING THE NAME OF A FUNCTION IS BEING USED.

ILLEGAL SUBLIST SPECIFICATION

-IN A DIMENSION STATEMENT, THERE ARE EITHER MORE THAN THREE SUBSCRIPTS ON AN ARRAY OR THE SUBSCRIPTS ARE NOT POSITIVE INTEGRAL CONSTANTS.

LAST STATEMENT NOT END

-EVERY FORTRN PROGRAM MUST HAVE AN END STATEMENT AS THE LAST STATEMENT. IT MUST START IN COLUMN SEVEN, AND MAY NOT HAVE SPACES IN IT.

MISSING LEFT PARENTHESIS

MISSING RIGHT PARENTHESIS

MISSING STRING DELIMITER

-A LITERAL FIELD IN A FORMAT STATEMENT IS MISSING A CLOSING APOSTROPHE.

NON-EXISTANT LINE REFERENCED

-A REFERENCE HAS BEEN MADE TO A LINE NUMBER WHICH DOES NOT EXIST OR THE PROGRAM ATTEMPTS TO BRANCH TO A NON-EXECUTABLE STATEMENT OR AN EXECUTABLE STATEMENT IS REFERENCED BY A READ OR WRITE STATEMENT.

NO STATEMENT TYPE FOUND

-THE STATEMENT TYPE REQUESTED IS EITHER NOT AVAILABLE OR HAS BEEN MISSPELLED.

UNDECIPHERABLE OPERAND

-AN ARITHMETIC EXPRESSION CANNOT BE EVALUATED IN THE MANNER WRITTEN. SOME CAUSES ARE FORGETTING THE MULTIPLICATION SIGN BETWEEN TWO QUANTITIES, OR USING THE WRONG OPERATORS. THE USER SHOULD CHECK THE MANUAL FOR THE CORRECT WAY OF CONSTRUCTING THE EXPRESSION.

EXECUTION ERRORS

BAD FLOATING POINT INPUT

-THE SYSTEM WAS UNABLE TO SUCCESSFULLY INTERPRET THE USER'S INPUT. COMMON CAUSES OF THIS ARE LEAVING OUT A DECIMAL POINT, INSERTING EXTRANEOUS CHARACTERS, OR MISMATCHING THE FORMAT SPECIFICATION WITH THE VARIABLE TYPE.

BAD INTEGER INPUT/OUTPUT

-THE SYSTEM WAS UNABLE TO SUCCESSFULLY INTERPRET THE USER'S INPUT. COMMON CAUSES OF THIS ARE INSERTING EXTRANEOUS CHARACTERS (INCLUDING DECIMAL POINTS), OR MISMATCHING THE FORMAT SPECIFICATION WITH THE VARIABLE TYPE, OR PRINTING A NEGATIVE NUMBER WITH AN I1 SPECIFICATION.

DO LOOPS NESTED TOO DEEP

-DO LOOPS WERE NESTED MORE THAN TEN DEEP.

END OF FILE DATA

-AN ATTEMPT WAS MADE TO READ AN UNDEFINED LOCATION IN THE FILE.

ILLEGAL ARITHMETIC PARAMETERS

-CAUSES OF THIS ERROR MESSAGE INCLUDE IMPROPER USE OF FUNCTIONS (SQRT OF A NEGATIVE NUMBER), DIVISION BY ZERO, OR RAISING ZERO TO A NEGATIVE POWER.

ILLEGAL EXPONENT

-THE EXPONENT ON A VALUE WHICH WAS READ USING AN E SPECIFICATION HAS AN EXPONENT WHICH IS 40 OR LARGER.

ILLEGAL FILE POINTER POSITION

-THE FILE POINTER HAS GONE OUT OF RANGE. THIS IS CAUSED BY BACKSPACING TO A POSITION BEFORE THE BEGINNING OF THE FILE, OR BY WRITING PAST THE PHYSICAL END OF THE FILE.

ILLEGAL LINE REFERENCED

-THE LAST STATEMENT OF A DO LOOP IS EITHER A DO, GOTO OR IF STATEMENT. THE PROGRAM MUST BE MODIFIED BEFORE IT CAN RUN.

ILLEGAL SUBSCRIPT

-THE SUBSCRIPT ON AN ARRAY IS EITHER LARGER THAN ITS DIMENSIONED VALUE OR IS NOT A POSITIVE INTEGER.

ILLEGAL USE OF COMPUTED GOTO

-THE INDEX VARIABLE IS EITHER GREATER THAN THE NUMBER OF STATEMENT NUMBERS LISTED, OR IS NOT A POSITIVE INTEGER.

MIXED MODE

-EXPRESSION CONTAINS MORE THAN ONE MODE WITHOUT APPROPRIATE MODE ALTERING FUNCTIONS.

NON-EXECUTABLE STATEMENT REFERENCED

-PROGRAM IS ATTEMPTING TO GO TO A NON-EXECUTABLE STATEMENT.

NON-EXISTANT FILE

-THE SYSTEM WAS UNABLE TO FIND THE REQUESTED FILE.

NON-FORMAT STATEMENT REFERENCED

-THE PROGRAM IS ATTEMPTING TO USE AN EXECUTABLE STATEMENT AS A FORMAT FOR READ OR WRITE.

OUT OF PROGRAM DATA

-AN ATTEMPT HAS BEEN MADE TO READ MORE DATA THAN IS LISTED AT END OF THE PROGRAM.

SAME VARIABLE IN TWO NESTED LOOPS

-PROGRAM HAS TWO NESTED LOOPS USING THE SAME INDEX VARAIBLE.

TOO MANY SUBSCRIPTED VARIABLES

-PROGRAM REQUESTS MORE THAN 200 LOCATIONS FOR SUBSCRIPTED VARAIBLES OR USES MORE THAN 12 DIFFERENT ARRAYS.

UNDEFINED VARIABLE ACCESSED

-USER HAS REQUESTED A VARIABLE WHICH HAS NOT BEEN PREVIOUSLY DEFINED BY THE PROGRAM.

```
.LIST
CIRCLES
C
C       CIRCLES
C       PROGRAM TO COMPUTE THE SIZE AND AREA OF THE INSCRIBED AND
C       CIRCUMSCRIBED CIRCLES OF A TRIANGLE GIVEN THE LENGTHS OF
C       THE SIDES OF THE TRIANGLE.
C       BY DAVE NESS
C
1       FORMAT(' THE DATA ',F4.1,',',F4.1,',';F4.1,' DOES NOT')
2       FORMAT(' DETERMINE A TRIANGLE.'//)
20      FORMAT(' FOR THE DATA ',F4.1,',';F4.1,',',F4.1)
3       FORMAT(' THE AREA OF THE TRIANGLE IS ',F8.5)
4       FORMAT(' THE RADIUS OF THE INSCRIBED CIRCLE IS ',F8.5)
5       FORMAT(' THE AREA OF THE INSCRIBED CIRCLE IS ',F8.5)
6       FORMAT(' THE RADIUS OF THE CIRCUMSCRIBED CIRCLE IS ',F8.5)
7       FORMAT(' THE AREA OF THE CIRCUMSCRIBED CIRCLE IS ',F9.5//)
8       FORMAT(3F4.1)
        DO 14 LOOP=1,4
        READ(2,8)SIDE1,SIDE2,HYPOT
        IF(SIDE1+SIDE2-HYPOT)9,9,10
9       WRITE(3,1)SIDE1,SIDE2,HYPOT
        WRITE(3,2)
        GOTO 14
10      IF(SIDE1)9,9,11
11      IF(SIDE2)9,9,12
12      IF(HYPOT)9,9,13
13      SIDE=.5*(SIDE1+SIDE2+HYPOT)
        AREA1=SQRT(SIDE*(SIDE-SIDE1)*(SIDE-SIDE2)*(SIDE-HYPOT))
        RADIU1=AREA1/SIDE
        AREA2=RADIU1**2.0*3.1415
        RADIU2=(SIDE1*SIDE2*HYPOT)/(4.0*AREA1)
        AREA3=RADIU2**2.0*3.1415
        WRITE(3,20)SIDE1,SIDE2,HYPOT
        WRITE(3,3)AREA1
        WRITE(3,4)RADIU1
        WRITE(3,5)AREA2
        WRITE(3,6)RADIU2
        WRITE(3,7)AREA3
14      CONTINUE
        STOP
        END
03.004.005.0
06.719.321.2
14.038.055.0
10.010.019.0
```

```
. EXECUTE


LOADING     FORTRAN 'CIRCLES'

EXECUTION   FORTRAN 'CIRCLES'


FOR THE DATA  3. 0, 4. 0, 5. 0
THE AREA OF THE TRIANGLE IS  6. 00000
THE RADIUS OF THE INSCRIBED CIRCLE IS  1. 00000
THE AREA OF THE INSCRIBED CIRCLE IS  3. 14150
THE RADIUS OF THE CIRCUMSCRIBED CIRCLE IS  2. 50000
THE AREA OF THE CIRCUMSCRIBED CIRCLE IS  19. 63438


FOR THE DATA  6. 7, 19. 3, 21. 2
THE AREA OF THE TRIANGLE IS 64. 15634
THE RADIUS OF THE INSCRIBED CIRCLE IS  2. 71849
THE AREA OF THE INSCRIBED CIRCLE IS 23. 21625
THE RADIUS OF THE CIRCUMSCRIBED CIRCLE IS 10. 68239
THE AREA OF THE CIRCUMSCRIBED CIRCLE IS 358. 48742


THE DATA 14. 0, 38. 0, 55. 0 DOES NOT
DETERMINE A TRIANGLE.


FOR THE DATA 10. 0, 10. 0, 10. 0
THE AREA OF THE TRIANGLE IS 43. 30127
THE RADIUS OF THE INSCRIBED CIRCLE IS  2. 88675
THE AREA OF THE INSCRIBED CIRCLE IS 26. 17916
THE RADIUS OF THE CIRCUMSCRIBED CIRCLE IS  5. 77350
THE AREA OF THE CIRCUMSCRIBED CIRCLE IS 104. 71664



EXECUTION COMPLETED
```

```
.LIST
FIBONACCI
C
C     FIBONACCI    A C/F DEMONSTRATION
C     PROGRAM TO COMPUTE AND PRINT THE FIRST 20 FIBONACCI NUMBERS
C     BY DAVE MENICOSY
C
100   FORMAT(I2,I15)
      DIMENSION IMP(20)
      IMP(1)=1
      IMP(2)=1
      DO 200 IMP1=3,20
      IMP2=IMP1-1
      IMP3=IMP1-2
200   IMP(IMP1)=IMP(IMP2)+IMP(IMP3)
      DO 300 IMP1=1,20
300   WRITE(3,100)IMP1,IMP(IMP1)
      STOP
      END
.EXECUTE


LOADING    FORTRAN 'FIBONACCI'

EXECUTION  FORTRAN 'FIBONACCI'


 1              1
 2              1
 3              2
 4              3
 5              5
 6              8
 7             13
 8             21
 9             34
10             55
11             89
12            144
13            233
14            377
15            610
16            987
17           1597
18           2584
19           4181
20           6765



EXECUTION COMPLETED
```

```
..LIST
PRIMES
C
C      PRIMES
C      PROGRAM TO FIND PRIME NNUMBERS FROM 1 TO 50
C      BY DAN CALEF
C
100    FORMAT(' THIS PROGRAM FINDS PRIME NUMBERS')
200    FORMAT(I3,'   IS PRIME')
       WRITE(3,100)
       ISTRT=2
       WRITE(3,200)ISTRT
       DO 99 ISTRT=3,50,2
       IWIRE=IFIX(SQRT(FLOAT(ISTRT)))
       DO 88 IZAP=3,IWIRE,2
       IF(ISTRT-ISTRT/IZAP*IZAP) 88,99,88
88     CONTINUE
       WRITE(3,200)ISTRT
99     CONTINUE
       STOP
       END
.EXECUTE


LOADING     FORTRAN 'PRIMES'

EXECUTION   FORTRAN 'PRIMES'


THIS PROGRAM FINDS PRIME NUMBERS
  2 IS PRIME
  3 IS PRIME
  5 IS PRIME
  7 IS PRIME
 11 IS PRIME
 13 IS PRIME
 17 IS PRIME
 19 IS PRIME
 23 IS PRIME
 29 IS PRIME
 31 IS PRIME
 37 IS PRIME
 41 IS PRIME
 43 IS PRIME
 47 IS PRIME



EXECUTION COMPLETED
```

```
.LIST
QUADRATIC
C
C      QUADRATIC       A C/F DEMONSTRATION
C      PROGRAM TO SOLVE QUADRATIC EQUATIONS
C      BY DAVE STARK
C
1      FORMAT(3F9.5)
2      FORMAT(' FOR THE EQUATION ',F9.5,'*X^2+',F9.5,'*X+',F9.5,'=0')
3      FORMAT(1H0)
5      FORMAT(F10.6,'   OR ',F10.6)
6      FORMAT(F10.6)
7      FORMAT(F6.4,' +',F6.4,'*I OR ',F7.4,'-',F5.3,'*I')
       DO 10 J=1,4
       WRITE(3,3)
       READ(2,1) A,B,C
       WRITE(3,2) A,B,C
       D=B**2.0-4.0*A*C
       IF(D)13,12,11
11     Q1=(SQRT(D)-B)/(2.0*A)
       Q2=((B+SQRT(D))*-1.0)/(2.0*A)
       WRITE(3,5) Q1,Q2
       GOTO 10
12     Q1=B/(2.0*A)*-1.0
       WRITE(3,6) Q1
       GOTO 10
13     Q1=B/(2.0*A)*-1.0
       Q2=SQRT(ABS(D))/(2.0*A)
       WRITE(3,7) Q1,Q2,Q1,Q2
10     CONTINUE
       STOP
       END
004.00000025.00000-21.00000
003.000000-1.70000007.00000
016.00000-24.00000009.00000
000.12000000.17341002.40000
.EXECUTE
```

```
LOADING    FORTRAN 'QUADRATIC'

EXECUTION  FORTRAN 'QUADRATIC'



FOR THE EQUATION    4.00000*X^2+ 25.00000*X+-21.00000=0
  0.750000 OR  -7.000000


FOR THE EQUATION    3.00000*X^2+ -1.70000*X+   7.00000=0
0.2833+1.5010*I OR  0.2833-1.501*I


FOR THE EQUATION  16.00000*X^2+-24.00000*X+   9.00000=0
  0.750000


FOR THE EQUATION    0.12000*X^2+  0.17341*X+   2.40000=0
-.7225+4.4134*I OR -0.7225-4.413*I


EXECUTION COMPLETED
```

```
.LIST
INVERSE
C
C       INVERSE              A C/F DEMONSTRATION
C       PROGRAM TO FIND MATRIX INVERSE
C       BY DAN CALEF AND DAVE MENICOSY
C
        DIMENSION A(10,20)
101     FORMAT(I2)
102     FORMAT(2F10.0)
12      FORMAT(' PIVOT ELEMENT =',E20.10)
61      FORMAT(1H#,F9.4)
62      FORMAT(1H )
        READ(1,101)N
        DO 200 I=1,N
        DO 200 J=1,N
200     READ(1,102) A(I,J)
        N1=2*N
        EPSIL=1.0E-8
        DO 3 I=1,N
        DO 3 J=1,N
        K=J+N
        IF(I-J)1,2,1
1       A(I,K)=0.0
        GOTO 3
2       A(I,K)=1.0
3       CONTINUE
        DO 50 IP=1,N
        IM=IP
        IST=IP+1
        DO 10 I=IST,N
        IF(ABS(A(IM,IP))-ABS(A(I,IP)))9,10,10
9       IM=I
10      CONTINUE
        IF(ABS(A(IM,IP))-EPSIL) 11,13,13
11      WRITE(3,12) A(IM,IP)
        IF(A(IM,IP))13,70,13
13      IF(IM-IP)14,20,14
14      DO 15 J=IP,N1
        AL=A(IP,J)
        A(IP,J)=A(IM,J)
15      A(IM,J)=AL
20      AL=A(IP,IP)
        A(IP,IP)=1.0
        DO 25 J=IST,N1
25      A(IP,J)=A(IP,J)/AL
        DO 40 I=1,N
        IF(I-IP) 30,40,30
30      AL=A(I,IP)
        DO 35 J=IP,N1
35      A(I,J)=A(I,J)-AL*A(IP,J)
40      CONTINUE
50      CONTINUE
        NP=N+1
```

```
      DO 60 I=1,N
      WRITE(3,62)
      DO 60 J=NP,N1
60    WRITE(3,61) A(I,J)
70    STOP
      END
.EXECUTE


LOADING     FORTRAN 'INVERSE'

EXECUTION   FORTRAN 'INVERSE'


*03
*2.
*3.
*7.
*5.
*3.
*-4.
*6.
*3.
*-9.

  -1.2500   4.0000  -2.7500
   1.7500  -5.0000   3.5833
  -0.2500   1.0000  -0.7500

EXECUTION COMPLETED

.
```

```
.LIST
CALENDER
C
C       CALENDER            A C/F DEMONSTRATION
C         PROGRAM CONVERTS THREE DIGIT NUMBER TO DAY IN THE YEAR
C       BY DAN CALEF
C
        DIMENSION M(12)
10      FORMAT(' THIS PROGRAM WILL CONVERT A THREE DIGIT NUMBER LESS')
20      FORMAT(' THAN 365 INTO A DAY OF THE YEAR.  ENTER THE THREE')
30      FORMAT(' DIGIT NUMBER THEN THE YEAR. (###,####)')
40      FORMAT(I3,1X,I4)
50      FORMAT(6I2)
60      FORMAT('  JANUARY ',I2,',  ',I4)
70      FORMAT('  FEBRUARY ',I2,',  ',I4)
80      FORMAT('  MARCH ',I2,',  ',I4)
90      FORMAT('  APRIL ',I2,',  ',I4)
100     FORMAT('  MAY ',I2,',  ',I4)
110     FORMAT('  JUNE ',I2,',  ',I4)
120     FORMAT('  JULY ',I2,',  ',I4)
130     FORMAT('  AUGUST ',I2,',  ',I4)
140     FORMAT('  SEPTEMBER ',I2,',  ',I4)
150     FORMAT('  OCTOBER ',I2,',  ',I4)
160     FORMAT('  NOVEMBER ',I2,',  ',I4)
170     FORMAT('  DECEMBER ',I2,',  ',I4)
180     FORMAT(' ANOTHER NUMBER AND YEAR (TYPE ZERO TO STOP)')
        READ(2,50)M(1),M(2),M(3),M(4),M(5),M(6)
        READ(2,50)M(7),M(8),M(9),M(10),M(11),M(12)
200     WRITE(3,10)
        WRITE(3,20)
        WRITE(3,30)
300     READ(1,40) NUMB,IYR
        IF(NUMB.EQ.0) GOTO 10000
        IF(MOD(IYR,4))320,330,320
330     M(2)=29
320     DO 400 I=1,12
        IF(NUMB.LT.M(I)) GOTO 500
        NUMB=NUMB-M(I)
400     CONTINUE
10000 STOP
500     IF(I.GT.6)GOTO 600
        GOTO(700,710,720,730,740,750)I
600     I=I-6
        GOTO(800,810,820,830,840,850)I
700     WRITE(3,60)NUMB,IYR
        GOTO 1000
710     WRITE(3,70)NUMB,IYR
        GOTO 1000
720     WRITE(3,80)NUMB,IYR
        GOTO 1000
730     WRITE(3,90)NUMB,IYR
        GOTO 1000
740     WRITE(3,100)NUMB,IYR
        GOTO 1000
```

```
750     WRITE(3,110)NUMB,IYR
        GOTO 1000
800     WRITE(3,120)NUMB,IYR
        GOTO 1000
810     WRITE(3,130)NUMB,IYR
        GOTO.1000
820     WRITE(3,140)NUMB,IYR
        GOTO 1000
830     WRITE(3,150)NUMB,IYR
        GOTO 1000
840     WRITE(3,160)NUMB,IYR
        GOTO 1000
850     WRITE(3,170)NUMB,IYR
1000    WRITE(3,180)
        M(2)=28
        GOTO 300
        END
312831303130
313130313031
.EXECUTE


LOADING     FORTRAN 'CALENDER'

EXECUTION   FORTRAN 'CALENDER'


THIS PROGRAM WILL CONVERT A THREE DIGIT NUMBER LESS
THAN 365 INTO A DAY OF THE YEAR.  ENTER THE THREE
DIGIT NUMBER THEN THE YEAR. (###,####)
*123,1973
 MAY  3, 1973
ANOTHER NUMBER AND YEAR (TYPE ZERO TO STOP)
*195,1973
 JULY 14, 1973
ANOTHER NUMBER AND YEAR (TYPE ZERO TO STOP)
*195,1972
 JULY 13, 1972
ANOTHER NUMBER AND YEAR (TYPE ZERO TO STOP)
*029,1973
 JANUARY 29, 1973
ANOTHER NUMBER AND YEAR (TYPE ZERO TO STOP)
*0


EXECUTION COMPLETED
```

```
.LIST
20?S
C
C       20?S
C       THE GAME OF TWENTY QUESTIONS
C       BY POBIN JEFFRIES
C
100     FORMAT(I2)
200     FORMAT(' TOO LOW')
201     FORMAT(' TOO HIGH')
300     FORMAT(' YES!!!!!!')
400     FORMAT(' THIS IS THE GAME OF 20 QUESTIONS.  I AM THINKING')
401     FORMAT(' OF A NUMBER FROM 1 TO 55.  CAN YOU GUESS WHAT IT IS?')
402     FORMAT(' FIRST TELL ME YOUR LUCKY SIX DIGIT NUMBER.')
403     FORMAT(I6)
600     FORMAT(' SORRY, YOU ARE OUT OF GUESSES.  THE NUMBER WAS',I3)
500     FORMAT(' WHAT IS YOUR GUESS?')
        WRITE(3,400)
        WRITE(3,401)
        WRITE(3,402)
        READ(1,403)NUM
        NUMBER=1
        DO 20 IMP=1,6
        I=NUM/10**(6-IMP)
        NUMBER=NUMBER+I
20      NUM=NUM-I*10**(6-IMP)
        DO 50 I=1,20
        WRITE(3,500)
        READ(1,100)IGUESS
        IF(IGUESS-NUMBER)40,60,45
45      WRITE(3,201)
        GOTO 50
40      WRITE(3,200)
50      CONTINUE
        WRITE(3,600)NUMBER
        STOP
60      WRITE(3,300)
        STOP
        END
```

```
..EXECUTE


LOADING   . FORTRAN '20?5'

EXECUTION  FORTRAN '20?5'


THIS IS THE GAME OF 20 QUESTIONS.   I AM THINKING
OF A NUMSER FROM 9 TO 55.   CAN YOU GUESS WHAT IT IS?
FIRST TELL ME YOUR LUCKY SIX DIGIT NUMBER.
*273524
WHAT IS YOUR GUESS?
*27
TOO HIGH
WHAT IS YOUR GUESS?
*20
TOO LOW
WHAT IS YOUR GUESS?
*23
YES!!!!!!


EXECUTION COMPLETED
```

```
.LIST
PREDIV
C
C     PREDIV        A C/F DEMONSTRATION
C     PROGRAM TO DIVIDE TWO NUMBERS TO 20 PLACES
C     BY DAN CALEF
C
99    FORMAT(' THIS PROGRAM DOES DIVISION TO 20 PLACES.')
98    FORMAT(' TYPE IN TWO NUMBERS (ONE IN THE FIRST SEVEN')
97    FORMAT(' SPACES, THE OTHER IN THE NEXT SEVEN SPACES.)')
100   FORMAT(2F7.2)
200   FORMAT(1H#//I6,'.')
300   FORMAT('#',I1)
      WRITE(3,99)
      WRITE(3,98)
      WRITE(3,97)
      READ(1,100) ALPHA,BETA
      IMP=INT(ALPHA/BETA)
      WRITE(3,200)IMP
      DO 77 INK=1,20
25    ALPHA=ALPHA-(FLOAT(IMP)*BETA)
      ALPHA=ALPHA*10.
35    IMP=INT(ALPHA/BETA)
      WRITE(3,300)IMP
77    CONTINUE
      STOP
      END
.EXECUTE


LOADING     FORTRAN 'PREDIV'

EXECUTION   FORTRAN 'PREDIV'


THIS PROGRAM DOES DIVISION TO 20 PLACES.
TYPE IN TWO NUMBERS (ONE IN THE FIRST SEVEN
SPACES, THE OTHER IN THE NEXT SEVEN SPACES.)
*000014.000017.


     0.82352941176470588235

EXECUTION COMPLETED
```

INDEX

A
A-STRING 1-12
ABS 1-11
AINT 1-11
ALOG 1-11
ALOG10 1-11
AMAX0 2-3
AMAX1 2-3
AMIN0 2-3
AMIN1 2-3
AMOD 2-3
ARITHMETIC OPERATORS 1-7
ARITHMETIC STATEMENT 1-7

B
^B 1-7, 3-2
BACKSPACE 2-7
BUG REPORTS PREFACE

C
NC-STRING 1-12
CARRIAGE CONTROL 2-4
CATALOG 1-6
COMMANDS 1-4 TO 1-6, 2-1
COMMENT 1-10
COMPUTED GOTO 2-5
CONTINUE 1-8
COS 1-11

D
ND 1-12
DATA FILES 2-6
DATA, PROGRAM 1-7, 1-2
DATA STATEMENT 2-4
DEFINE FILE 2-6, 1-2
DEVICE CODES 1-7, 1-8
DIM 2-3
DIMENSION 2-5
DO 1-8, 2-2

E
EDITING 1-12
END 1-10, 1-2, 2-2
ERASE 1-6
ERRORS 3-5 TO 3-8
 SYNTAX AND PROGRAM CONSTRUCTION 3-5, 3-6
 EXECUTION ERRORS 3-7, 3-8
EXECUTE 1-4
NEW.D 1-9
EXP 1-11

F
FILE DECLARATION 2-6
FILE READ 2-6
FILE SIZES 1-1, 2-6, 3-1
FILE WRITE 2-6
FLOAT 1-11
FORMAT 1-9, 2-4
FORMAT SPECIFICATIONS 1-9, 2-4
FUNCTIONS 1-11, 2-3
NFW.D 1-9

G
GOTO 1-8, 2-5
GROUP USE 3-4

H
HEIRARCHY 1-7
HELP 1-5
HOLLERTIH FIELD 2-4
NHTEXT 2-4

I
NI-STRING 1-12
IABS 1-11
IDIM 2-3
IF, MULTIBRANCH 1-10, 2-2
IF, LOGICAL 2-2
IFIX 1-11
INT 1-11
INTEGER 2-2
ISIGN 2-3
NIW 1-9

K
KJOB 1-5

L
LIST 1-5
LN COMMAND 1-12, 2-1
LOAD 1-4
LOGICAL IF 2-2
LOADING INSTRUCTIONS 3-3, 3-4
LOOPS 1-8

M
MAXIMUM PROGRAM SIZE 1-1, 3-1
MAX0 2-3
MAX1 2-3

MIN0 2-3
MIN1 2-3
MOD 2-3
MODE OF EXPRESSIONS 1-7
MULTIBRANCH GOTO 2-5

O
OPEN COMMAND 2-1

P
PAUSE 2-3
PROGRAM DATA 1-7, 1-2
PROGRAMS, SAMPLE SECTION 4
PUNCH 2-1

R
READ 1-7, 2-6
REASSIGN 2-1
REAL 2-2
RETRIEVE 1-6
REWIND 2-7

S
SAMPLE PROGRAMS SECTION 4
SIGN 2-3
SIN 1-11
SPECIAL CHARATERS 3-2
SORT 1-11
STOP 1-10
STORE 1-6
SUBSCRIPTED VARIABLES 2-5
SYSTEM AND GROUP LIBRARY USE
 2000B/E 1-1, 3-3
 2000C/F 3-4
SYSTEM OPERATION 1-1

T
TABBING 1-2, 1-3
TAPE 1-4
TAPE LOADING INSTRUCTIONS
 2000B/E 1-1, 3-3
 2000C/F 3-4
TPUNCH 1-5

U
UJOB 1-5

V
VARIABLES 1-3

W
WRITE 1-8, 2-6

Y
^Y 1-4

OMSI 2000 C/F FORTRAN SUMMARY

COMMANDS

Command	Purpose
LOAD	Accept a FORTRAN program
AI (ctrl I)	Tab over line number field
AY (ctrl Y)	Exit from LOAD command
EXECUTE	Execute FORTRAN program
LIST	Print text of program
PUNCH	Punch tape of program
TPUNCH	Punch tape of program, using AI for tabbing
TAPE	Accept program from tape
KJOB	Return to BASIC, deleting current program
UJOB	Return to BASIC, without deleting program
HELP	Summary of system operation

EDITING COMMANDS

LN-##1,##2	Lists program with sequence numbers from the line ##1 to ##2. "LN" gives whole text.
###C-text	Changes sequencd numbered line ### to "text"
###I-text	Inserts "text" before line ###
###D	Deletes line ###
A-text	Makes "text" final line of the program

CATALOG COMMANDS

CA	Catalog of programs in user's program storage file
ST	Stores current program in user's storage file
R name	Retrieves program "name"
E name	Erases program "name"
O name,<len>	Creates a file "name" which can hold <len> numbers
RA	Reassign another program storage file

FUNCTIONS AND OPERATORS

Functions

REAL argument, REAL value

SIN(rx)	sine of rx in radians
COS(rx)	cosine of rx
SQRT(rx)	square root
ABS(rx)	absolute value
EXP(rx)	e to power rx
AINT(rx)	truncation
ATAN(rx)	arctan in radians
ALOG(rx)	log base e
ALOG10(rx)	log base 10
AMOD(r1,r2)	remainder of r1/r2
SIGN(r1,r2)	transfer of sign
DIM(r1,r2)	positive difference
AMAX1(r1,r2,...)	largest value
AMIN1(r1,r2,...)	smallest value

REAL argument, INTEGER value

IFIX(rx)	real to integer
INT(rx)	real to integer
MAX1(r1,r2,...)	largest value
MIN1(r1,r2,...)	smallest value

INTEGER argument, REAL value

FLOAT(ix)	integer to real
AMAX0(i1,i2,...)	largest value
AMIN0(i1,i2,...)	smallest value

INTEGER argument, INTEGER value

IABS(ix)	absolute value
MOD(i1,i2)	remainder of i1/i2
ISIGN(i1,i2)	transfer of sign
IDIM(i1,i2)	positive difference
MAX0(i1,i2,...)	largest value
MIN0(i1,i2,...)	smallest value

Operators

+	addition	.EQ.	=	.NE.	≠
-	subtraction	.GT.	>	.LT.	<
*	multiply	.GE.	≥	.LE.	≤
/	divide			.OR.	logical or
**	exponentiate			.AND.	logical and
		.EOR.	exclusive or		

STATEMENTS

Statement form	Remarks
<var>=<exp>	Stores value of <exp> in <var>
C comment	A "C" in the first column causes rest of the line to be ignored.
CONTINUE	No operation, dummy statement
DATA <var>,<var>,.../<number>,<number>.../	Sets <var>s to corresponding <number>s. Non-executable.
DEFINE FILE (<prog stor file>,<file>)	Defines which file in which program storage file will be used for file commands. Non-executable.
DIMENSION <var>(<bounds>),...	Reserves space for arrays. Maximum of 200 variables in 12 arrays. Non-executable.
DO ### <ivar>=<initial>,<ifinal>,<istep>	Causes lines down to and including ### to be repeated 1+(<ifinal>-<initial>)/<istep> times. <istep> default is +1.
END	Indicates end of text.
FORMAT(<spec>,<spec>,...)	Formats input-output. Specifications allowed are nIw - Integer nFw.d - Floating point nEw.d - Exponent form nX - Space 'text' - Literal field wHtext - Hollerith field where "n" is the replicator, "w" is the field width, "d" is the number of digits after the decimal point, and "text" is a string. The first character of the first string is used for carriage control.
GOTO ###	Transfer control to line with line number ###
GOTO (##1,##2,...)<ivar>	If <ivar> equals n, control goes to ##n.
IF (<exp>) ##1,##2,##3	Branches depending on value of <exp>; to ##1, if <exp> < 0 to ##2, if <exp> = 0 to ##3, if <exp> > 0
IF (<exp>) <statement>	Logcal IF. Executes statement if <exp> is true (≠0)
INTEGER <var>,<var>,...	Defines <var> to be an integer variable. Non-executable
PAUSE <number>	Causes 5 second delaly, printing out <number>, if present.
READ (1,###) <var list>	Reads values for variables from the terminal, using FORMAT statement ###.
READ (2,###) <var list>	Reads values for variables from data, using FORMAT statement ###. Everything after the END is data.
READ (4) <var list>	Reads values for variables from file. File definded by DEFINE FILE.
REAL <var>,<var>,...	Defines <var> to be a real variable. Non-executable.
STOP	Halts execution.
WRITE (3,###) <var list>	Prints values of variables out on terminal using FORMAT statement ###.
WRITE (5) <var list>	Writes values of variables onto the file. File defined by DEFINE FILE.

Variables and operations

Variables may have 1 to 6 characters. The first must be a letter, others may be letters or digits. If the first letter is a I, J, K, L, M, or N the variable will be an integer. Starting with any other first letter will make the variable floating point. All operations must take place between two quantities of the same mode. All arrays must be dimensioned, and may have 1 to 3 dimensions. The mode of the entire array is determined by its first character. Only integer value or integer variables may be used as subscripts.